

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicants: Choi, *et al.*

Examiner: Wei, Zheng

Serial No.: 10/790,913

Group Art Unit: 2192

Filed: 03/02/2004

Docket No.: **END920040013US1**

Title: **PORTLET TEMPLATE BASED ON A STATE DESIGN PATTERN**

---

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**OFFICE ACTION RESPONSE**

Sir:

This communication is in response to the Final Office Action mailed February 20, 2007.

i

**In the Claims:**

Please cancel claims 5, 14, and 23. Please amend claims 3-4, 6-9, 12-13, 15-18, 21-22, and 24-27. The claims are as follows:

1-2. (Canceled)

3. (Currently amended) A method for designing object-oriented software for implementing portlets of a portal, said portlets adapted to be available to a user of the software during a session in which the user clicks on a link of a first page to identify an action object of an Action class and a state object of a State class, said method comprising designing the software followed by storing the designed software in a computer usable medium from which the software may be executed on a processor of a computer system to implement portlets of a portal, said designing the software comprising:

including in the software a Portlet Template that comprises the State class, the Action class, and program code;

including in the State class a perform view method for displaying a view of a page;

including in the Action class an action performed method for performing an action and a set state method for setting the state object into the session;

adapting the program code to execute:

(a) the action performed method of the action object to perform the action,

(b) the set state method of the action object to set the state object into the session,

and

(c) the perform view method of the state object to display a view of a second page that is associated with the action;

including a portlet code module and a controller code module in the program code, wherein the portlet code module and a controller code module are different modules, wherein the portlet code module is adapted to execute the action performed method and the set state method, and wherein the controller code module is adapted to execute the perform view method; [[and]]

including a BasePortlet class and a TemplatePortlet class in the portlet code module, wherein the BasePortlet class does not include a portlet specific method and includes the action performed method and the set state method, and wherein the TemplatePortlet class is a child of the BasePortlet class and includes at least one portlet specific method; and

including a BaseController class and a TemplateControllerForHtml class in the controller code module,

wherein the BaseController class does not include a portlet specific method and includes the perform view method,

wherein the TemplateControllerForHtml class is a child of the BaseController class and includes at least one portlet specific method, and

wherein the BasePortlet class and the BaseController class are different classes.

4. (Currently amended) The method of claim 3, wherein said designing the software further comprises including an action listener method in the BasePortlet class, wherein the action object resulting from the user clicking on the link of the first page is communicated from the portal to the action listener method in response to the action listener method extracting the action object

from an input parameter associated with the link, wherein the extracted action object includes an action performed method for performing an action and a set state method for setting a destination of the State class into the session, and wherein the destination state includes a display method to be executed for displaying a next page following the action.

5. (Canceled)

6. (Currently amended) The method of claim 3, wherein a first object of the State class includes a first perform view method for displaying a first portlet state of a given page, wherein a second object of the State class includes a second perform view method for displaying a second portlet state of the given page, [[and]] wherein the first and second portlet states are different portlet states, and wherein the first and second perform view methods are different perform view methods.

7. (Currently amended) The method of claim 6,

wherein the first and second portlet states are each selected from the group consisting of a Normal portlet state, a Maximized portlet state, and a Minimized portlet state;

wherein in the Normal portlet state, a portlet of the portlets is arranged on a displayed portal page with other portlets of the portlets;

wherein in the Maximized portlet state, the portlet of the portlets is displayed in the entire body of the displayed portal page, replacing a view of the other portlets; and

wherein in the Minimized portlet state, only a portlet title bar is displayed on the

displayed portal page with the other portlets.

8. (Currently amended) The method of claim 3, wherein the software comprises Java JAVA software.

9. (Currently amended) The method of claim 3, wherein the portal comprises a ~~WSP~~ portal server.

10-11. (Canceled)

12. (Currently amended) A computer system comprising a processor and a computer readable memory unit coupled to the processor, said memory unit containing object-oriented software adapted to be executed by the processor to implement portlets of a portal,

said portlets adapted to be available to a user of the software during a session in which the user clicks on a link of a first page to identify an action object of an Action class and a state object of a State class;

said software including a Portlet Template that includes the State class, the Action class, and program code;

said State class comprising a perform view method for displaying a view of a page;

said Action class comprising an action performed method for performing an action and a set state method for setting the state object into the session;

said program code adapted to execute:

(a) the action performed method of the action object to perform the action,

(b) the set state method of the action object to set the state object into the session,

and

(c) the perform view method of the state object to display a view of a second page that is associated with the action,

wherein the program code includes a portlet code module and a controller code module,

wherein the portlet code module and a controller code module are different modules,

wherein the portlet code module is adapted to execute the action performed method and the set state method,

wherein the controller code module is adapted to execute the perform view method,

wherein the portlet code module comprises a BasePortlet class and a TemplatePortlet class,

wherein the BasePortlet class does not include a portlet specific method and includes the action performed method and the set state method, [[and]]

wherein the TemplatePortlet class is a child of the BasePortlet class and includes at least one portlet specific method,

wherein the controller code module comprises a BaseController class and a

TemplateControllerForHtml class,

wherein the BaseController class does not include a portlet specific method and includes the perform view method,

wherein the TemplateControllerForHtml class is a child of the BaseController class and includes at least one portlet specific method, and

wherein the BasePortlet class and the BaseController class are different classes.

13. (Currently amended) The computer system of claim 12, wherein the BasePortlet class includes an action listener method, and wherein the action object resulting from the user clicking on the link of the first page is communicated from the portal to the action listener method in response to the action listener method extracting the action object from an input parameter associated with the link, wherein the extracted action object includes an action performed method for performing an action and a set state method for setting a destination of the State class into the session, and wherein the destination state includes a display method to be executed for displaying a next page following the action.

14. (Canceled)

15. (Currently amended) The computer system of claim 12, wherein a first object of the State class includes a first perform view method for displaying a first portlet state of a given page, wherein a second object of the State class includes a second perform view method for displaying a second portlet state of the given page, [[and]] wherein the first and second portlet states are different portlet states, and wherein the first and second perform view methods are different perform view methods.

16. (Currently amended) The computer system of claim 15,

wherein the first and second portlet states are each selected from the group consisting of a

Normal portlet state, a Maximized portlet state, and a Minimized portlet state;

wherein in the Normal portlet state, a portlet of the portlets is arranged on a displayed portal page with other portlets of the portlets;

wherein in the Maximized portlet state, the portlet of the portlets is displayed in the entire body of the displayed portal page, replacing a view of the other portlets; and

wherein in the Minimized portlet state, only a portlet title bar is displayed on the displayed portal page with the other portlets.

17. (Currently amended) The computer system of claim 12, wherein the software comprises ~~Java~~ JAVA software.

18. (Currently amended) The computer system of claim 12, wherein the portal comprises a ~~WSP~~ portal server.

19-20. (Canceled)

21. (Currently amended) A computer program product, comprising a computer usable medium having computer readable object-oriented software embodied therein, said computer readable object-oriented software containing program code that when executed by a processor of a computer system implements portlets of a portal,

said portlets adapted to be available to a user of the software during a session in which the user clicks on a link of a first page to identify an action object of an Action class and a state



object of a State class;

said software including a Portlet Template that includes the State class, the Action class, and program code;

said State class comprising a perform view method for displaying a view of a page;

said Action class comprising an action performed method for performing an action and a set state method for setting the state object into the session;

said program code adapted to execute:

(a) the action performed method of the action object to perform the action,

(b) the set state method of the action object to set the state object into the session,

and

(c) the perform view method of the state object to display a view of a second page that is associated with the action,

wherein the program code includes a portlet code module and a controller code module,

wherein the portlet code module and a controller code module are different modules,

wherein the portlet code module is adapted to execute the action performed method and the set state method,

wherein the controller code module is adapted to execute the perform view method,

wherein the portlet code module comprises a BasePortlet class and a TemplatePortlet class,

wherein the BasePortlet class does not include a portlet specific method and includes the action performed method and the set state method, and

wherein the TemplatePortlet class is a child of the BasePortlet class and includes at least

one portlet specific method,

wherein the controller code module comprises a BaseController class and a  
TemplateControllerForHtml class,

wherein the BaseController class does not include a portlet specific method and includes  
the perform view method,

wherein the TemplateControllerForHtml class is a child of the BaseController class and  
includes at least one portlet specific method, and

wherein the BasePortlet class and the BaseController class are different classes.

22. (Currently amended) The computer program product of claim 21, wherein the BasePortlet class includes an action listener method, and wherein the action object resulting from the user clicking on the link of the first page is communicated from the portal to the action listener method in response to the action listener method extracting the action object from an input parameter associated with the link, wherein the extracted action object includes an action performed method for performing an action and a set state method for setting a destination of the State class into the session, and wherein the destination state includes a display method to be executed for displaying a next page following the action.

23. (Canceled)

24. (Currently amended) The computer program product of claim 21, wherein a first object of the State class includes a first perform view method for displaying a first portlet state of a given

page, wherein a second object of the State class includes a second perform view method for displaying a second portlet state of the given page, [[and]] wherein the first and second portlet states are different portlet states, and wherein the first and second perform view methods are different perform view methods.

25. (Currently amended) The computer program product of claim 24,

wherein the first and second portlet states are each selected from the group consisting of a Normal portlet state, a Maximized portlet state, and a Minimized portlet state;

wherein in the Normal portlet state, a portlet of the portlets is arranged on a displayed portal page with other portlets of the portlets;

wherein in the Maximized portlet state, the portlet of the portlets is displayed in the entire body of the displayed portal page, replacing a view of the other portlets; and

wherein in the Minimized portlet state, only a portlet title bar is displayed on the displayed portal page with the other portlets.

26. (Currently amended) The computer program product of claim 21, wherein the software comprises Java JAVA software.

27. (Currently amended) The computer program product of claim 21, wherein the portal comprises a WSP portal server.

## REMARKS

**Applicants respectfully request that the present office action mailed 02/20/2007 be changed from a final office action to a non-final office action, based on Applicants' arguments presented *infra*.**

The Examiner objected to claim 20.

The Examiner rejected claims 8-9, 17-18 and 26-27 under 35 U.S.C. § 112, second paragraph, as being indefinite for allegedly failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

The Examiner rejected claims 3-9, 12-18 and 21-27 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Hanis (Hanis *et al.*, "Applying the State Pattern to WebSphere Portal Portlets", Part 1- Overview, Part2 - Implementation, 12/11/2002) in view of Jacobson (Jacobson *et al.*, Object-Oriented Software Engineering, A User Case Driven Approach. - Art is now being made of record.)

The Examiner rejected claims 7, 16 and 25 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Hanis (Hanis *et al.*, "Applying the State Pattern to WebSphere Portal Portlets", Part 1- Overview, Part2 - Implementation, 12/11/2002) in view of Jacobson (Jacobson *et al.*, Object-Oriented Software Engineering, A User Case Driven Approach) in further view of Hepper (Hepper *et al.*, "Introducing the Portlet Specification, Part 1", 08/01/2003).

Applicants respectfully traverse the finality of the present office action mailed 02/20/2007, the claim objection, the § 112 rejections and § 103 rejections with the following arguments.

**Finality of Office Action Mailed 02/20/2007**

Applicants respectfully assert that the finality of the present office action mailed 02/20/2007 is improper in light of MPEP 706.07(a) which recites: “Under present practice, second or any subsequent actions on the merits shall be final, except where the examiner introduces a new ground of rejection that is neither necessitated by applicant's amendment of the claims nor based on information submitted in an information disclosure statement filed during the period set forth in 37 CFR 1.97(c) ...”.

Applicants note that claim 12 was rejected on a new ground (Hanis in view of Jacobson under 35 U.S.C. § 103(a)) not necessitated by Applicants' amendment of claim 12. Claim 12 was amended in Applicants' prior office action response as rewritten in independent form to include all of the limitations of claims 10 and 11 from which claim 12 had depended, resulting in claim 12 being otherwise the same claim 12 as was originally filed. Therefore, the rejection of claim 12 on the new ground was not necessitated by Applicants' amendment of claim 12.

In addition, claims 8, 17, and 26 were rejected on a new ground (35 U.S.C. § 112, second paragraph) not necessitated by Applicants' amendment of claims 8, 17, and 26-27, because the new ground for rejecting claims 8, 17, and 26 involves the same issue that was present in the originally filed claims 8, 17, and 26. The rejection of claims 8, 17, and 26 on a new ground pertaining to “Java” being a trademark was not necessitated by Applicants' amendment of claims 8, 17, and 26.

In addition, claims 9, 18, and 27 were rejected on a new ground (35 U.S.C. § 112, second paragraph) not necessitated by Applicants' amendment of claims 9, 18, and 27-27, because the new ground for rejecting claims 9, 18, and 27 involves the same issue that was present in the

originally filed claims 9, 18, and 27. The rejection of claims 9, 18, and 27 on a new ground pertaining to the term “WSP server” requiring definition was not necessitated by Applicants’ amendment of claims 9, 18, and 27.

Accordingly, Applicants respectfully request that the the present office action mailed 02/20/2007 be changed from a final office action to a non-final office action.

**Claim Objection**

The Examiner objected to claim 20. The Examiner argues: "Claims 20 has been canceled by Applicant, but it is still marked "Original" at page 11. It should be deleted or crossed over."

In response, Applicants have deleted the text of claim 20, and claim 20 is properly marked as deleted.

**35 U.S.C. § 112, Second Paragraph**

The Examiner rejected claims 8-9, 17-18 and 26-27 under 35 U.S.C. § 112, second paragraph, as being indefinite for allegedly failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

**Claims 8, 17, and 26**

The Examiner argues: “The trademarks "Java" are used in a claims as a limitation to identify or describe a particular material or product, the claims do not comply with the requirements of the 35 U.S.C. 112, second paragraph. Ex parte Simpson, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product. To expedite correction on this matter, the Examiner suggests amend the term "Java software" to -- "JAVA software" —”.

In response, Applicants have amended claims 8, 17, and 26 to replace “Java software” with “JAVA software”, as suggested by the Examiner.

**Claims 9, 18, and 27**

The Examiner argues: “The term "WSP server" has to be defined in the claim to properly identify any particular material or product. The Examiner suggests adding definition of WSP to the claim.”

In response, Applicants have amended claims 9, 18, and 27 to replace “WSP server” with “portal server”, which eliminates “WSP server” in claims 9, 18, and 27.



Accordingly, Applicants respectfully request that the rejections of claims 8-9, 17-18 and 26-27 under 35 U.S.C. § 112, second paragraph be withdrawn.

**35 U.S.C. § 103(a): Claims 3-9, 12-18 and 21-27**

The Examiner rejected claims 3-9, 12-18 and 21-27 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Hanis (Hanis *et al.*, “Applying the State Pattern to WebSphere Portal Portlets”, Part 1- Overview, Part2 - Implementation, 12/11/2002) in view of Jacobson (Jacobson *et al.*, Object-Oriented Software Engineering, A User Case Driven Approach. - Art is now being made of record.)

Since claims 5, 14, and 23 have been canceled, the rejection of claims 5, 14, and 23 under 35 U.S.C. § 103(a) is moot.

Applicants respectfully contend that claims 3, 12, and 21 are not unpatentable over Hanis in view of Jacobson, because Hanis in view of Jacobson does not teach or suggest each and every feature of claims 3, 12, and 21.

As a first example of why claims 3, 12, and 21 are not unpatentable over Hanis in view of Jacobson, Hanis in view of Jacobson does not teach or suggest the feature of including: a portlet code module including a BasePortlet class, a controller code module including a BaseController class, wherein the portlet code module and a controller code module are different modules, and wherein the BasePortlet class and the BaseController class are different classes.

The Examiner has not identified, in Hanis, different modules respectively representing the portlet code module and the controller code module.

Moreover, the Examiner relies on the StateManagerPortlet class in Hanis being both the BasePortlet class and the BaseController class, which conflicts with the feature of the BasePortlet class and the BaseController class being different classes. The Examiner has not identified, in

Hanis, different classes respectively representing the BasePortlet class and the BaseController class.

Therefore, Hanis in view of Jacobson does not teach or suggest the preceding feature of claims 3, 12, and 21.

As a second example of why claims 3, 12, and 21 are not unpatentable over Hanis in view of Jacobson, Hanis in view of Jacobson does not teach or suggest the feature: “wherein the BasePortlet class ... includes the action performed method and the set state method ...; ... wherein the BaseController class ... includes the perform view method, ... and wherein the BasePortlet class and the BaseController class are different classes.”

In contrast, the Examiner argues that Hanis teach that a single class, namely the StateManagerPortlet class, includes the action performed method, the set state method, and the perform view method.

Therefore, Hanis in view of Jacobson does not teach or suggest the preceding feature of claims 3, 12, and 21.

As a third example of why claims 3, 12, and 21 are not unpatentable over Hanis in view of Jacobson, Hanis in view of Jacobson does not teach or suggest the feature: “wherein the BasePortlet class does not include a portlet specific method ..., ... wherein the BaseController class does not include a portlet specific method ..., and wherein the BasePortlet class and the BaseController class are different classes.”

Applicants assert that neither Hanis nor Jacobson teaches the preceding feature of claims

3, 12, and 21. The Examiner has cited anything in Hanis or Jacobson that allegedly teaches the preceding feature of claims 3, 12, and 21.

Moreover, Hanis **teaches away** from the preceding feature of claims 3, 12, and 21 by reciting (part 1, page 2): “StateManagerPortlet ...is the main portlet class ... where you typically write ***all of*** the portlet-specific code” (emphasis added).

Therefore, Hanis in view of Jacobson does not teach or suggest the preceding feature of claims 3, 12, and 21.

As a fourth example of why claims 3, 12, and 21 are not unpatentable over Hanis in view of Jacobson, Hanis in view of Jacobson does not teach or suggest the feature: “wherein the TemplatePortlet class is a child of the BasePortlet class and includes at least one portlet specific method; ... wherein the TemplateControllerForHtml class is a child of the BaseController class and includes at least one portlet specific method, and wherein the BasePortlet class and the BaseController class are different classes.”

The Examiner argues:

“StateManagerPortlet implements the actionPerformed method...” but does not explicitly disclose StateManagerPortlet class can be implemented as base-child class architecture wherein generic methods can be defined in base class and the portlet specific methods can be defined in extended child class. However, Jacobson in the same analogous art of Object-Oriented software engineering discloses the basic feature of object-oriented programming – Inheritance (see for example, Fig.3.14 and related text, also see p.58, lines 19-25, "By means of extracting and sharing common characteristics, we can generalize classes and place them higher up in an inheritance hierarchy. In the same way, if we wish to

add a new class, we can find a class that already offers some of the operations and information structure required for the new class. We can then let the new class inherit this class and only add anything which is unique for the new class. We then specialize the class").

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use this inheritance feature to define generic and common features of StateManagerProtlet class in a base class (for example, Baseportlet class) and to define specific feature (for example, Templateportet class with porlet specific method) in the child class of the base class. In this way, base class can be reused to derive and generate different specific classes. One would have been motivated to use inheritance feature to reuse common description and make it very easy to modify the code as suggested by Jacobson (see for example, p.57, line 9-10, "Hence we can reuse common descriptions", also see p.58, lines 7-8, "Hence inheritance is very useful for easy modification of models")".

In response, Applicants that the Examiner's citation of Jacobson merely teaches class inheritance generically, which is well known. The Examiner has not cited any reference to support the Examiner's contention that it is obvious to modify Hanis to incorporate the specific parent-child class structure recited in claims 3, 12, and 21. The Examiner offers two reasons why it is allegedly obvious to so modify Hanis: (1) to derive and generate different specific classes; and (2) to make it very easy to modify the code.

As to the Examiner's first argument (1) that modifying Hanis is obvious in order to derive and generate different specific classes, Applicants note that Hanis has already derived and generated the specific classes needed to implement Hanis' methodology (i.e.,

StateManagerPortlet class, action classes, and state classes). The Examiner has not identified any other classes that Hanis' methodology would allegedly require.

As to the Examiner's second argument (2) that modifying Hanis is obvious in order to make it very easy to modify the code, Applicants first note that "the code" lacks antecedent basis in that the Examiner has not identified which code "the code" relates to. Therefore, the Examiner's second argument is ambiguous and cannot be reasonably understood.

In addition with respect to the Examiner's second argument, Applicants respectfully point out that Hanis' stated implementation specifically enables highly simplified code. See Hanis (part 1, page 2): "Therefore, the StateManagerPortlet does not need to know anything about the current portlet implementation and does not have a large number of ifs and checks to determine where processing should continue. As long as you are familiar with the flow between states and actions, the processing will occur properly, **without you having to code much extraneous control logic**" (emphasis added).

Moreover, Hanis teaches away from being modified by the features of "wherein the TemplatePortlet ... includes at least one portlet specific method; ... wherein the TemplateControllerForHtml class ... includes at least one portlet specific method", because Hanis recites (part 1, page 2): "StateManagerPortlet ... is the main portlet class ... where you typically write *all of* the portlet-specific code" (emphasis added). The methodology of Hanis is based on *all of* the portlet-specific code being comprised by StateManagerPortlet. Therefore, to modify Hanis such that portlet-specific code is not comprised by StateManagerPortlet **would destroy** Hanis' methodology.

Therefore, Applicants respectfully maintain that the Examiner's argument for modifying

Hanis by the cited teaching of Jacobson is not persuasive. Accordingly, Hanis in view of Jacobson does not teach or suggest the preceding feature of claims 3, 12, and 21.

Based on the preceding arguments, Applicants respectfully maintain that claim 3 is not unpatentable over Hanis in view of Jacobson, and that claim 3 is in condition for allowance. Since claims 4 and 6-9 depend from claim 3, Applicants contend that claims 4-9 are likewise in condition for allowance. Since claims 13 and 15-18 depend from claim 12, Applicants contend that claims 13-18 are likewise in condition for allowance. Since claims 22 and 24-27 depend from claim 21, Applicants contend that claims 22-27 are likewise in condition for allowance.

In addition with respect to claims 4, 13, and 22, Applicants maintain that Hanis in view of Jacobson does not teach or suggest the feature: “wherein the action object resulting from the user clicking on the link of the first page is communicated from the portal to the action listener method in response to the action listener method extracting the action object from an input parameter associated with the link, wherein the extracted action object includes an action performed method for performing an action and a set state method for setting a destination of the State class into the session, and wherein the destination state includes a display method to be executed for displaying a next page following the action”.

The Examiner cites Hanis (Part 1, pages 4, part 2, pages 6-7, implementation examples).

In response, Applicants cannot find in Hanis (Part 1, pages 4, part 2, pages 6-7, implementation examples) a teaching or suggestion of claims 4, 13, and 22

Therefore, claims 4, 13, and 22 are not unpatentable over Hanis in view of Jacobson.

In addition with respect to claims 6, 15, and 24, Applicants maintain that Hanis in view of Jacobson does not teach or suggest the feature: “wherein a first object of the State class includes a first perform view method for displaying a first portlets state of a given page, wherein a second object of the State class includes a second perform view method for displaying a second portlets state of the given page, and wherein the first and second portlets states are different portlets states”.

The Examiner argues: “Hanis and Jacobson disclose the method of claim 3 and Hanis and Jacobson disclose the method of claim 3 wherein a first object of the State class includes a first performView method for displaying a first portlet state of a given page, wherein a second object of the State class includes a second performView method for displaying a second portlet state of the given page, and wherein the first and second portlet states are different portlet states (Part1, page 2, "Typically, the state's perform method will invoke a JSP to render its results", part2, page 14-15, section "Wrapping up the portlet states and actions")”.

In response, Applicants respectfully contend that the Examiner’s citation to Hanis (Part 1, page 2, "Typically, the state's perform method will invoke a J.P. to render its results") does not teach or suggest the claimed first perform view method and second perform view method subject to the claimed limitations on the first perform view method and second perform view method (including the limitation of the first and second perform view methods are different perform view methods).

In further response, Applicants respectfully contend that the Examiner’s citation to Hanis (part 2, page 14-15, section "Wrapping up the portlet states and actions") specifically recites:



“The remaining part of the application follows this same pattern. The process control logic flows exactly the same way for the Edit mode as it does for the View mode.

1. The initial state for the Edit view is provided through the InitialStateManager class.
2. The State class's perform view method is invoked, gets its needed model objects, and passes them to the J.P.
3. The J.P. assigns one or more PortletURIs to actions on the UI, and associates each with a specific action class.
4. When the user clicks on a link, the action class's action performed method is invoked. The action logic is performed and the appropriate state is set.
5. Processing continues in this manner as the user navigates through the portlet”.

Applicants respectfully contend that the preceding citation to Hanis (part 2, page 14-15, section "Wrapping up the portlet states and actions") does not teach the claimed first perform view method and second perform view method subject to the claimed limitations on the first perform view method and second perform view method (including the limitation of the first and second perform view methods are different perform view methods).

Therefore, claims 6, 15, and 24 are not unpatentable over Hanis in view of Jacobson.

**35 U.S.C. § 103(a): Claims 7, 16, and 25**

The Examiner rejected claims 7, 16 and 25 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Hanis (Hanis *et al.*, “Applying the State Pattern to WebSphere Portal Portlets”, Part 1- Overview, Part2 - Implementation, 12/11/2002) in view of Jacobson (Jacobson *et al.*, Object-Oriented Software Engineering, A User Case Driven Approach) in further view of Hepper (Hepper *et al.*, “Introducing the Portlet Specification, Part 1”, 08/01/2003).

Since claims 7, 16 and 25 respectively depend from claims 3, 12, and 21, which Applicants have argued *supra* to not be unpatentable over Hanis in view of Jacobson under 35 U.S.C. §103(a), Applicants maintain that claims 7, 16 and 25 are likewise not unpatentable over Hanis in view of Jacobson and further in view of Hepper under 35 U.S.C. §103(a).

In addition with respect to claims 7, 16 and 25, Applicants maintain that Hanis in view of Jacobson and further in view of Hepper does not teach or suggest the feature:

“wherein in the Normal portlet state, a portlet of the portlets is arranged on a displayed portal page with other portlets of the portlets;

wherein in the Maximized portlet state, the portlet of the portlets is displayed in the entire body of the displayed portal page, replacing a view of the other portlets; and

wherein in the Minimized portlet state, only a portlet title bar is displayed on the displayed portal page with the other portlets.”

The Examiner argues: “Hanis and Jacobson ... do not explicitly disclose that the portlet states are selected from the group consisting of a Normal portlet state, a Maximized portlet state, and a Minimized portlet state. However, Hepper discloses that the Portlet Specification – JSR 168 defines the following window states: Normal, Maximized and Minimized. (Pages 5-6,

Window states). It would have been obvious to one having ordinary skill in the art at the time the invention was made to implement those three Portlet Specification defined states. One would have been motivated and required to implement Normal, Maximized and Minimized state defined by the Portlet Specification, because, these are required features in JSR 168 according to Hepper.”

In response, Applicants notes that JSR 168 defines Normal, Maximized and Minimized portlet states, and Applicants acknowledge the definition of Normal portlet state in JSR 168 discloses the features of claimed Normal portlet state. However, JSR 168 does not disclose the features of the claimed Maximized portlet state and the claimed Minimized portlet state (recited *supra*) as evidenced by the following definitions in JSR 168 of the Maximized and Minimized portlet states as recited on page 7 of Hepper:

“Maximized: Indicates that a portlet may be the only portlet on the portal page or that the portlet has more space compared to other portlets in the portal page, and can therefore produce richer content than in a normal window state...

Minimized: Indicates that the portlet should only render minimal output or no output at all.”

In addition, Applicants respectfully contend that the Examiner’s argument as to why it is allegedly obvious to modify Hanis in view of Jacobson by the preceding definitions in Hepper is not persuasive. The Examiner’s argument for the modification of Hanis in view of Jacobson is that the Normal, Maximized and Minimized state are required features in JSR 168.

In response, Applicants note that all that th Examiner’s citation of JSR 168 (as described by Hepper) contains no more than definitions of the Normal, Maximized, and Minimized states

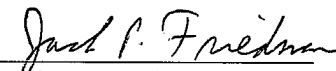
and does not state anywhere that the Normal, Maximized, and Minimized states are required. Therefore, Applicants respectfully maintain that the Examiner's argument for modifying Hanis in view of Jacobson by the cited teaching of Hepper is not persuasive.

Accordingly, Hanis in view of Jacobson and further in view of Hepper does not teach or suggest the preceding feature of claims 7, 16 and 25.

### CONCLUSION

Based on the preceding arguments, Applicants respectfully believe that all pending claims and the entire application meet the acceptance criteria for allowance and therefore request favorable action. If the Examiner believes that anything further would be helpful to place the application in better condition for allowance, Applicants invites the Examiner to contact Applicants' representative at the telephone number listed below. The Director is hereby authorized to charge and/or credit Deposit Account 09-0457.

Date: 04/09/2007

  
\_\_\_\_\_  
Jack P. Friedman  
Registration No. 44,688

Schmeiser, Olsen & Watts  
22 Century Hill Drive - Suite 302  
Latham, New York 12110  
(518) 220-1850